```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE GENCOMER
OBJECT MODULE PLACED IN GENCOM.OBJ
COMPILER INVOKED BY:  PLM80 GENCOM.PLM DEBUG OPTIMIZE PAGEWIDTH(132)


           $ TITLE('CPM 3.0 --- GENCOM 1.0')
           /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                * * * GENCOM * * *

            * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 1         gencomer:
           do;


 2  1      declare
              mpmproduct literally '01h', /* requires mp/m */
              cpmversion literally '30h'; /* requires 3.0 cp/m */


 3  1      declare plm label public;

 4  1      declare copyright (*) byte data (
          ' Copyright (c) 1982, Digital Research ');

 5  1      declare version (*)     byte data('11/02/82');

           /*
                     Digital Research
                     Box 579
                     Pacific Grove, Ca
                     93950
           */
```

CP/M plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
          $ eject
          /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                              * * *  CP/M INTERFACE * * *

           * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 6  1     declare
              maxb      address external,     /* addr field of jmp BDOS */
              fcb (33)  byte external,        /* default file control block */
              fcb16(33) byte external,        /* default fcb 2 */
              buff(128) byte external,        /* default buffer */
              buffa     literally '.buff',    /* default buffer */
              fcba      literally '.fcb',     /* default file control block */

              cr            literally '13',
              lf            literally '10';

                                              /* reset drive mask */
 7  1        declare reset$mask (16) address data (
               0000000000000001b,
               0000000000000010b,
               0000000000000100b,
               0000000000001000b,
               0000000000010000b,
               0000000000100000b,
               0000000001000000b,
               0000000010000000b,
               0000000100000000b,
               0000001000000000b,
               0000010000000000b,
               0000100000000000b,
               0001000000000000b,
               0010000000000000b,
               0100000000000000b,
               1000000000000000b );

 8  1     mon1: procedure(f,a) external;
 9  2        declare f byte, a address;
10  2        end mon1;

11  1     mon2: procedure(f,a) byte external;
12  2        declare f byte, a address;
13  2        end mon2;

14  1     declare mon3 literally 'mon2a';

15  1     mon3: procedure(f,a) address external;
16  2        declare f byte, a address;
17  2        end mon3;

          /********** SYSTEM FUNCTION CALLS **********************/

18  1     printchar: procedure(char);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 19  2      declare char byte;
 20  2      call mon1(2,char);
 21  2  end printchar;

 22  1  printb: procedure;
                                    /* print blank character */
 23  2      call printchar(' ');
 24  2  end printb;

 25  1  printx: procedure(a);
 26  2      declare a address;
 27  2      declare s based a byte;
 28  2          do while s <> 0;
 29  3                  call printchar(s);
 30  3                  a = a + 1;
 31  3          end;
 32  2  end printx;

 33  1  check$con$stat: procedure byte;
 34  2      return mon2(11,0);          /* console ready */
 35  2  end check$con$stat;

 36  1  crlf: procedure;
 37  2      call printchar(cr);
 38  2      call printchar(lf);
 39  2      if check$con$stat then do;
 41  3          call mon1 (1,0);        /* read character */
 42  3          call mon1 (0,0);        /* system reset */
 43  3      end;
 44  2  end crlf;

 45  1  print: procedure(a);
 46  2      declare a address;
                            /* print the string starting at address a until the
                               next 0 is encountered */
 47  2      call crlf;
 48  2      call printx(a);
 49  2  end print;

 50  1  get$version: procedure address;
                            /* returns current cp/m version # */
 51  2      return mon3(12,0);
 52  2  end get$version;


 53  1  conin: procedure byte;
 54  2      return mon2(6,0fdh);
 55  2  end conin;


 56  1  open: procedure(fcb) byte;
 57  2      declare fcb address;
 58  2      return mon2(15,fcb);
 59  2  end open;

 60  1  close: procedure(fcb) byte;
 61  2          declare fcb     address;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 62  2          return mon2(16,fcb);
 63  2      end close;

 64  1      make: procedure(fcb) byte;
 65  2          declare fcb     address;
 66  2          return mon2(22,fcb);
 67  2      end make;

 68  1          declare ioflag  address,
                     nrecs  byte;

 69  1      mread: procedure(fcb);          /* multi sector read - returns # recs*/
 70  2          declare fcb     address;

 71  2          ioflag = mon3(20,fcb);
 72  2          readflag = low(ioflag);         /* if = 255 then error */
 73  2          nrecs = high(ioflag);           /* if 0 -> multi sector count */

 74  2      end mread;


 75  1      setmulti: procedure(nsects);        /* set multi sector count */
 76  2          declare nsects  byte;

 77  2          flag = mon2(44,nsects);

 78  2      end setmulti;


 79  1      readsq: procedure(fcb) byte;
 80  2          declare fcb     address;
 81  2          return mon2(20,fcb);
 82  2      end readsq;

 83  1      writesq: procedure(fcb) byte;
 84  2          declare fcb     address;
 85  2          return mon2(21,fcb);
 86  2      end writesq;

 87  1      rename: procedure(fcb) byte;
 88  2          declare fcb     address;
 89  2          return mon2(23,fcb);
 90  2      end rename;

 91  1      delete: procedure(fcb) byte;
 92  2          declare fcb     address;
 93  2          return mon2(19,fcb);
 94  2      end delete;

 95  1      setdma: procedure(dma);
 96  2         declare dma address;
 97  2         call mon1(26,dma);
 98  2      end setdma;

 99  1      return$errors:                  /* Off => return BDOS errors */
             procedure(mode);
100  2         declare mode byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
101   2          call moni (45,mode);
102   2      end return$errors;

             /*************************************************************/

103   1      terminate: procedure;
104   2          call crlf;
105   2          call moni (0,0);
106   2      end terminate;

107   1      parse: procedure(pfcb) address external;
108   2              declare pfcb address;

109   2      end parse;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
            $eject

110  1              declare

                        options(*) byte data
                                   ('NULLOLOADEROSCB',0FFH),
                        off$opt(*) byte data(0,5,12,15),
                        end$list       byte data (0ffh),
                        end$of$string  byte data (0),

                        delimiters(*) byte data (0,'[]=, :;<>%\|"()/*!@$+-*?',0,0ffh),
                        SPACE          byte data(5),   /* delim       space */
                        COMMA          byte data(4),   /*    '        comma */
                        LPAREN         byte data(14),  /*    '        left paren */

                        opt$map(23)    byte,

                        j              byte initial(0),
                        buf$ptr        address,
                        opt$index      byte,
                        endbuf         byte,
                        delimiter      byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
$ eject

111  1          declare
                    true            literally '1',
                    false           literally '0',
                    punchSCB        byte initial (false),
                    COMonly         byte initial (false),
                    revert          byte initial (false),
                    build           byte initial (false),
                    replace         byte initial (false),
                    empty           byte initial (false),
                    hex             byte initial (false),

                    oldSCB          byte initial (false),

                    incount         byte initial (0),
                    ret$inst        byte data (0c9h),
                    BLANK           byte data (020h),
                    (readflag,writeflag)    byte,
                    flag            byte,
                    (rsx,old,fill)  byte,
                    maxrcd          byte data(32),

                    deletes         byte,
                    which(15)       byte,

                    comoff          address,
                    comsize         address,
                    totbyte         address,
                    rsxrec          address,
                    oldrsx          address,
                    offsets(15)     address,
                    length$rsx(15)  address,
                    testvers        address,

                    comtype(3)      byte data ('COM'),
                    hextype(3)      byte data ('HEX'),
                    rsxtype(3)      byte data ('RSX'),

                    tempfcb(33)     byte initial(0,'TEMP    $$$',0,0,0,0,0),
                    errfcb(14)      byte,

                  files(16)       structure ( pass(8) byte),
                    len$pass(16)    byte,

                    parse$struc     structure(
                       name$addr    address,
                       fcb$addr     address),

                    optmark         based buf$ptr byte,
                    NULL            byte initial(0),
                    LOAD            byte initial(0),
                    SCB             byte initial(0),

                    fcbs(16)        structure(
                       file(33)       byte),
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                test$ptr        address,
                allfcbs(16)     address,
                fcbp            address,
                comptr          address,
                comfcb          based comptr (1) byte,
                testfcb         based test$ptr (1) byte,
                gen$fcb         based fcbp (1) byte;

        /*      RSX COM FILE HEADER FORMAT              */

112  1      declare
                head$ptr        address,
                head            based head$ptr structure(
                 retinst        byte,   /* return instruction 0C9h */
                 progsize       address,/* program size:orig com prog */
                 SCBjmp         byte,
                 SCBaddr        address,
                 RESERVED2(7)   byte,
                 LOADER         byte,
                 nscb           byte,
                 nrsx           byte);  /* number of RSX modules in file */

113  1      declare
                subptr          address,
                rsx$sub$head    based subptr structure(
                 off            address,
                 len            address,
                 NONBANK        byte,
                 RESERVED3      byte,
                 name(8)        byte,
                 RESERVED4      address),

                scbvect         based subptr structure(
                  pad1          byte,
                  smark         byte,
                  pad2          address,
                  svect(12)     byte),

                head$byte       based head$ptr byte,

                head$buffer(384)        byte,
                iobuff(4096)            byte,

                nextptr         address,
                next            based nextptr structure(
                 off            address,
                 len            address,
                 NONBANK        byte,
                 RESERVED3      byte,
                 name(8)        byte,
                 RESERVED4      address),

                nbank(16)       byte initial(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
                newoff(16)      address,
                newlen(16)      address,
                actlen(15)      address,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        new(15)         structure(
                          name(8)       byte),

                        soff(20)        byte,
                        sval(20)        byte,
                        nscbs           byte initial(0);

114  1          declare
                        SCBbuff(256)    byte,
                        SCBcode(23)     byte data(011h,018h,00,0d5h,0eh,031h,0cdh,5,0,
                                                  0e1h,23h,23h,23h,7eh,0feh,
                                                  0ffh,0e5h,0ebh,0c2h,4,0,0e1h,0c9h),
                        SCBpos          address;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
          $eject

115  1          declare
                      ERRORM(*)              byte data ('ERROR: ',0),
                      FILEM(*)               byte data ('FILE: ',0),
                      err$notfnd(*)          byte data ('File not found.',0),
                      err$msg$make(*)        byte data ('No directory space.',0),
                      err$msg$parse(*)       byte data ('Invalid file name.',0),
                      err$msg$first(*)       byte data ('First submitted file must be

          a COM file.',0),
                      err$msg$dup1(*)        byte data ('Duplicate input RSX...',0),
                      err$msg$dup2(*)        byte data ('Duplicate RSX in header.',
                                                        ' Replacing old by new.',0),

                      err$msg$rsxval(*)      byte data ('Invalid RSX type.',0),
                      err$msg$no$rsx(*)      byte data ('No more RSX files to be used

          .',0),
                      err$msg$copy(*)        byte data ('Error on copy.',0),
                      err$msg$rsx$slot(*)    byte data ('There are not enough availab

          le RSX slots.',0),
                      err$msg$read(*)        byte data ('Disk read.',0),
                      err$msg$write(*)       byte data ('Disk write.',0),
                      err$msg$toobig(*)      byte data ('Total file size exceeds 64K.

          ',0),
                      err$NULL(*)     byte data ('COM file found and NULL option.',0),

                      errSTRIP(*)     byte data ('No header or RSXs to strip.',0),

                      errIFCB(*)      byte data ('Invalid FCB.',0),
                      errMEDIA(*)     byte data ('Media change occurred.',0),
                      errDIO(*)       byte data ('Disk I/O error.',0),
                      errDRIVE(*)     byte data ('Invalid drive error.',0),

                      errscboff(*)    byte data ('Invalid SCB offset',0),
                      errscbclose(*)  byte data('Missing right parenthesis.',0),
                      errscbnoval(*)  byte data ('Missing SCB value.',0),
                      errscbpar(*)    byte data ('Missing left parenthesis.',0),
                      err$unrecopt(*) byte data ('Unrecognized option.',0),
                      err$notscb(*)   byte data ('No modifier for this option.',0);

116  1     closeall: procedure;
117  2          declare i      byte;

118  2          do i = 0 to incount;
119  3               readflag = close(allfcbs(i));   /* close input files */
120  3          end;
121  2          readflag = close(.tempfcb);
122  2          readflag = delete(.tempfcb);

123  2     end closeall;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
124  1     get$errfcb: procedure;
125  2           declare (i,j)  byte;

126  2           do i = 1 to 14;
127  3                 errfcb(i) = 0;
128  3           end;
129  2           errfcb(0) = 9;                /* tab */

130  2           i = 1;
131  2           j = 1;
132  2           do while i < 9 and gen$fcb(j) <> 32;          /* 32 = space */
133  3                 errfcb(i) = gen$fcb(j);
134  3                 i = i + 1;
135  3                 j = j + 1;
136  3           end;

137  2     ge1:  errfcb(i) = 46;               /* dot */
138  2           j = 9;
139  2           do while i < 12 and gen$fcb(j) <> 32;
140  3                 i = i + 1;
141  3                 errfcb(i) = gen$fcb(j);
142  3                 j = j + 1;
143  3           end;
144  2     end get$errfcb;


145  1     e$print1: procedure(message);
146  2           declare message address;

147  2           call get$errfcb;
148  2           call print(.ERRORM);
149  2           call printx(message);

150  2     end e$print1;

151  1     e$print2: procedure;

152  2           call print(.FILEM);
153  2           call printx(.errfcb);
154  2           call crlf;

155  2     end e$print2;


156  1     err$print: procedure(message);
157  2           declare message address;

158  2           call e$print1(message);
159  2           call e$print2;

160  2           call closeall;
161  2           call terminate;

162  2     end err$print;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
163  1     bdoserr: procedure;
164  2             declare (lflag,hflag)   byte;

165  2             lflag = low(ioflag);
166  2             hflag = high(ioflag);

167  2             if lflag = 9 then call err$print(.errIFCB);
169  2             if lflag = 10 then call err$print(.errMEDIA);
171  2             if lflag = 255 then do;
173  3                     if hflag = 1 then call err$print(.errDIO);
175  3                     if hflag = 4 then call err$print(.errDRIVE);
177  3             end;

178  2     end bdoserr;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

$ eject

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
          $eject
          /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                              * * * Option scanner  * * *

           * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

179  1    separator: procedure(character) byte;

                                                  /* determines if character is a
                                                     delimiter and which one */
180  2          declare k       byte,
                        character       byte;

181  2          k = 1;
182  2    loop: if delimiters(k) = end$list then return(0);
184  2          if delimiters(k) = character then return(k);    /* null = 25 */
186  2                k = k + 1;
187  2                go to loop;

188  2    end separator;

189  1    opt$scanner:    procedure(list$ptr,off$ptr,idx$ptr);
                                                  /* scans the list pointed at by idxptr
                                                     for any strings that are in the
                                                     list pointed at by list$ptr.
                                                     Offptr points at an array that
                                                     contains the indices for the known
                                                     list. Idxptr points at the index
                                                     into the list. If the input string
                                                     is unrecognizable then the index is
                                                     0, otherwise > 0.

                                                  First, find the string in the known
                                                  list that starts with the same first
                                                  character.  Compare up until the next
                                                  delimiter on the input. if every input
                                                  character matches then check for
                                                  uniqueness.  Otherwise try to find
                                                  another known string that has its first
                                                  character match, and repeat.  If none
                                                  can be found then return invalid.

                                                  To test for uniqueness, start at the
                                                  next string in the known list and try
                                                  to get another match with the input.
                                                  If there is a match then return invalid.

                                                  else move pointer past delimiter and
                                                  return.

                                                P.Balma        */

190  2          declare
```

COPYRIGHT © 1932
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        buff            based buf$ptr (1) byte,
                        idx$ptr         address,
                        off$ptr         address,
                        list$ptr        address;

191  2          declare
                        i               byte,
                        j               byte,
                        list            based list$ptr (1) byte,
                        offsets         based off$ptr (1) byte,
                        wrd$pos         byte,
                        character       byte,
                        letter$in$word  byte,
                        found$first     byte,
                        start           byte,
                        index           based idx$ptr byte,
                        save$index      byte,
                        (len$new,len$found)     byte,
                        valid           byte;

           /*****************************************************************************/
           /*                      internal subroutines                                 */
           /*****************************************************************************/

192  2     check$in$list: procedure;
                                /* find known string that has a match with
                                   input on the first character.  Set index
                                   = invalid if none found.  */

193  3          declare i       byte;

194  3          i = start;
195  3          wrd$pos = offsets(i);
196  3          do while list(wrd$pos) <> end$list;
197  4                  i = i + 1;
198  4                  index = i;
199  4                  if list(wrd$pos) = character then return;
201  4                  wrd$pos = offsets(i);
202  4          end;
                                /* could not find character */
203  3          index = 0;
204  3          return;
205  3     end check$in$list;

206  2     setup:  procedure;
207  3          character = buff(0);
208  3          call check$in$list;
209  3          letter$in$word = wrd$pos;
                                /* even though no match may have occurred, position
                                   to next input character.  */
210  3          i = 1;
211  3          character = buff(1);
212  3     end setup;

213  2     test$letter:    procedure;
                                /* test each letter in input and known string */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #:

```
214  3          letter$in$word = letter$in$word + 1;

                                        /* too many chars input? 0 means
                                           past end of known string */
215  3          if list(letter$in$word) = end$of$string then valid = false;
                else
217  3          if list(letter$in$word) <> character then valid = false;

                i = i + 1;
220  3          character = buff(i);

221  3     end test$letter;

222  2     skip:  procedure;
                                        /* scan past the offending string;
                                           position buf$ptr to next string...
                                           skip entire offending string;
                                           ie., falseopt=mod, [note: comma or
                                           space is considered to be group
                                           delimiter] */
223  3          character = buff(i);
224  3          delimiter = separator(character);
225  3          do while ((delimiter <> 2) and (delimiter <> 4) and (delimiter <> 5)
                          and (delimiter <> 25));
226  4               i = i + 1;
227  4               character = buff(i);
228  4               delimiter = separator(character);
229  4          end;
230  3          endbuf = i;
231  3          buf$ptr = buf$ptr + endbuf + 1;
232  3          return;
233  3     end skip;

234  2     eat$blanks: procedure;

235  3          declare charac  based buf$ptr byte;


236  3          do while(delimiter != separator(charac)) = SPACE;
237  4               bufptr = buf$ptr + 1;
238  4          end;

239  3     end eat$blanks;

      /*************************************************************************/
      /*                      end of internals                                 */
      /*************************************************************************/


                                        /* start of procedure */
240  2          call eat$blanks;
241  2          start = 0;
242  2          call setup;

                                        /* match each character with the option
                                           for as many chars as input
                                           Please note that due to the array
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                                        indices being relative to 0 and the
                                        use of index both as a validity flag
                                        and as a index into the option/mods
                                        list, index is forced to be +1 as an
                                        index into array and 0 as a flag*/

243  2          do while index <> 0;
244  3                  start = index;
245  3                  delimiter = separator(character);

                                        /* check up to input delimiter */

246  3                  valid = true;          /* test$letter resets this */
247  3                  do while delimiter = 0;
248  4                         call test$letter;
249  4                         if not valid then go to exit1;
251  4                         delimiter = separator(character);
252  4                  end;

253  3                  go to good;

                                        /* input <> this known string;
                                           get next known string that
                                           matches */
254  3     exit1:       call setup;
255  3           end;
                                        /* fell through from above, did
                                           not find a good match*/
256  2           endbuf = i;            /* skip over string & return*/
257  2           call skip;
258  2           return;

                                        /* is it a unique match in options
                                           list? */
259  2     good: endbuf = i;
260  2           len$found = endbuf;
261  2           save$index = index;
262  2           valid = false;
263  2     next$opt:
                    start = index;
264  2           call setup;
265  2           if index = 0 then go to finished;

                                        /* look at other options and check
                                           uniqueness */

267  2           len$new = offsets(index + 1) - offsets(index) - 1;
268  2           if len$new = len$found then do;
270  3                  valid = true;
271  3                  do j = 1 to len$found;
272  4                         call test$letter;
273  4                         if not valid then go to next$opt;
275  4                  end;
276  3           end;
277  2           else go to nextopt;
                                        /* fell through...found another valid
                                           match --> ambiguous reference */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER.

```
278  2            index = 0;
279  2            call skip;              /* skip input field to next delimiter*/
280  2            return;

281  2      finished:                     /* unambiguous reference */
                  index = save$index;
282  2            buf$ptr = buf$ptr + endbuf;
283  2            call eat$blanks;
284  2            if delimiter <> 0 then  buf$ptr = buf$ptr + 1;
286  2            else delimiter = SPACE;

287  2      end opt$scanner;

288  1      error$prt:      procedure;
289  2            declare i       byte,
                       t       address,
                       char    based t byte;

290  2            t = buf$ptr - endbuf - 1;
291  2            do i = 1 to endbuf;
292  3                  call printchar(char);
293  3                  t = t + 1;
294  3            end;

295  2      end error$prt;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
            $eject

296   1     e$print3: procedure(message);

297   2           declare message address;

298   2           call print(.ERRORM);
299   2           call printx(message);
300   2           call terminate;

301   2     end e$print3;


302   1     aschex: procedure(ahbyte,albyte) byte;

303   2           declare (ahbyte,albyte) address,
                        hbyte           based ahbyte byte,
                        lbyte           based albyte byte;

304   2      conv: procedure(abyte);
305   3           declare abyte   address,
                        b       based abyte byte;

306   3           if b > 39h then b = b - 37h;
308   3           else b = b - 30h;

309   3      end conv;

310   2           call conv(ahbyte);
311   2           call conv(albyte);
312   2           hbyte = shl(hbyte,4);

313   2           return(hbyte or lbyte);

314   2     end aschex;

            /*****************************************************************************/

315   1     valoff: procedure(high,low,achar);
316   2           declare (high,low)      byte,
                        achar           address,
                        char            based achar byte;

317   2           if (char > high) or (char < low) then
318   2                 call e$print3(.errscboff);

319   2     end valoff;

            /*****************************************************************************/

            /*****************************************************************************/

320   1     getoption: procedure;

321   2           declare char            based buf$ptr byte,
                        bufptri         address,
                        nextchar        based bufptri byte,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        index           byte,
                        zero            byte;

                /*************************************************/

322   2     getscbval: procedure;

323   3             bufptr1 = buf$ptr + 1;

324   3             if (delimiter := separator(nextchar)) = 0 then do;
326   4                     sval(nscbs) = aschex(buf$ptr,buf$ptr1);         /* 2 chars */
327   4                     buf$ptr = buf$ptr + 2;
328   4             end;
329   3             else do;
330   4                     sval(nscbs) = aschex(.zero,buf$ptr);            /* 1 char */
331   4                     buf$ptr = bufptr1;
332   4             end;

333   3             nscbs = nscbs + 1;

334   3             if (delimiter := separator(char)) <> 15 then            /* ) */
335   3                     call e$print3(.errscbclose);

336   3             buf$ptr = buf$ptr + 1;

337   3             delimiter = separator(char);                    /* set delimiter */
338   3             if delimiter <> 0 then buf$ptr = buf$ptr + 1;

340   3     end getscbval;

                /*******************************************************/

341   2     checkval: procedure;

342   3             delimiter = separator(char);
343   3             if delimiter = SPACE then go to cv0;
345   3             if delimiter <> COMMA then
346   3                     call e$print3(.err$scbnoval);

347   3     cv0:    buf$ptr = buf$ptr + 1;

348   3     end checkval;

                /*******************************************************/


349   2     getscboff: procedure;

350   3             if (delimiter := separator(char)) = LPAREN then do;

352   4                     buf$ptr = buf$ptr + 1;
353   4                     call valoff(39h,30h,buf$ptr);           /* valid char ? */

354   4                     bufptr1 = buf$ptr + 1;

355   4                     delimiter = separator(nextchar);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SFR. #

```
356  4                  if delimiter = SPACE then go to gs1;
358  4                  if delimiter = COMMA then go to gs1;
                                                        /* 2 char input */
360  4                        call valoff(36h,30h,buf$ptr);
361  4                        call valoff(46h,30h,bufptr1);   /* valid ? */
362  4                        soff(nscbs) = aschex(buf$ptr,bufptr1);
363  4                        buf$ptr = buf$ptr + 2;
364  4                        call checkval;
365  4                        return;

                                                        /* single char in */
366  4     gs1:               soff(nscbs) = aschex(.zero,buf$ptr);
367  4                        buf$ptr = bufptr1 + 1;
368  4           end;
369  3           else call e$print3(.errscbpar);

370  3     end getscboff;

           /*********************************************************/

371  2           zero = 30h;
372  2           delimiter = 1;
373  2           index = 0;
374  2           buf$ptr = buf$ptr + 1;          /* move off [ delimiter */

                                                /* while not eos */

375  2     gto0: call opt$scanner(.options,.off$opt,.index);
376  2           if index = 0 then do;
378  3                 call print(.ERRORM);
379  3                 call printx(.err$unrecopt);
380  3                 call print(.('OPTION: ',0));
381  3                 call error$prt;
382  3           end;

383  2           if index = 1 then NULL = true;
385  2           else if index = 2 then LOAD = true;

                 if delimiter = 2 then return;
389  2           if delimiter = 25 then return;

391  2                 if delimiter = 3 then do;             /* = */
393  3                       if index <> 3 then do;
395  4                             call print(.ERRORM);
396  4                             call printx(.err$notscb);
397  4                             call opt$scanner(.options,.offopt,
                                                      .index);
398  4                             go to gto1;
399  4                       end;

400  3                       call getscboff;         /* buf$ptr -> value */
401  3                       call getscbval;
402  3                       SCB = true;
403  3                 end;

404  2     gto1: if delimiter = 0 then return;
406  2           if delimiter = 2 then return;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
408  2            if delimiter = 25 then return;

410  2            go to gto0;

411  2      end getoption;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
            $ eject


412  1      opener: procedure(fcb);
413  2              declare fcb     address;

414  2              if open(fcb) > 3 then do;
416  3                      fcbp = fcb;
417  3                      call err$print(.err$notfnd);
418  3              end;

419  2      end opener;


420  1      closer: procedure(fcb);
421  2              declare fcb     address;

422  2              if close(fcb) > 3 then do;
424  3                      fcbp = fcb;
425  3                      call err$print(.err$notfnd);
426  3              end;
427  2      end closer;

428  1      maker: procedure(fcb);
429  2              declare fcb     address;

430  2              flag = make(fcb);
431  2              if flag > 3 then do;
433  3                      fcbp = fcb;
434  3                      call err$print(.err$msg$make);
435  3              end;

436  2      end maker;

437  1      deleter: procedure;

438  2              if (comfcb(8) and 80h) = 80h then return;       /* user 0 file ? */

440  2              if delete(comptr) > 0 then do;
442  3                      fcbp = comptr;
443  3              end;

444  2      end deleter;


445  1      parser: procedure(fcb$ptr);

446  2              declare fcb$ptr address;

447  2              parse$struc.name$addr = buf$ptr;
448  2              parse$struc.fcb$addr = fcb$ptr;
449  2              test$ptr = buf$ptr;

450  2      pa1:    buf$ptr = parse(.parse$struc);  /* parse command tail */

451  2      pa2:    if buf$ptr = 0ffffh then do;
453  3                      fcbp = test$ptr;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
454  3                  call err$print(.err$msg$parse);
455  3          end;

456  2     end parser;


457  1     copypass$dma: procedure(index);
458  2          declare index   byte,
                     i       byte;

459  2          do i = 0 to 7;
460  3               buff(i) = files(index).pass(i);
461  3          end;

462  2     end copypass$dma;

463  1     renamer: procedure;

464  2          declare
                     (i,j)           byte,
                     renbuf(32)      byte;

465  2          do i = 12 to 15;
466  3               j = i + 16;
467  3               renbuf(i) = 0;
468  3               renbuf(j) = 0;
469  3          end;

470  2          do i = 0 to 11;         /* set up buffer */
471  3               j = i + 16;
472  3               renbuf(i) = tempfcb(i);
473  3               renbuf(j) = comfcb(i);
474  3          end;

475  2     rei:   flag = rename(.renbuf);

476  2          if flag > 0 then do;
478  3               fcbp = allfcbs(0);              /*GLITCH??????????*/
479  3          end;
480  2     end renamer;

481  1     clearfcb: procedure(fcb);

482  2          declare fcb     address,
                     f       based fcb (1) byte,
                     i       byte;

483  2          do i = 12 to 33;
484  3               f(i) = 0;
485  3          end;

486  2     end clearfcb;


          /*****************************************************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
487  1     copy: procedure(recsize);
488  2           declare recsize              address;
489  2           declare recs                 based recsize address;
490  2           declare
                        i                    byte,
                        flag                 address;

491  2           call setmulti(maxrcd);
492  2           call mread(fcbp);

493  2     co2:  if readflag <> 0 then do;
495  3                 if readflag = 1 then do;
497  4                    if nrecs = 0 then  return;    /* EOF */
499  4                 end;
500  3                 else call bdoserr;
501  3           end;

502  2           i = maxrcd;
503  2           if nrecs <> 0 then do;          /* read less than maxrcd */
505  3                 call setmulti(nrecs);
506  3                 i = nrecs;
507  3           end;

508  2           writeflag = writesq(.tempfcb);

509  2           do while i <> 0;
510  3                 recs = recs + 128;      /* this is in bytes */
511  3                 i = i - 1;
512  3           end;
                                              /* record count <= 64K */
513  2           if recs > 0ffffh then call err$print(.err$msg$toobig);

515  2           if nrecs <> 0 then return;

517  2           call mread(fcbp);

518  2           go to co2;

519  2     end copy;


      /*****************************************************************************/


520  1     copy2: procedure(nrcds,skip);
                                        /* read/write in min(maxrcd,nrcds)
                                           units. */

521  2           declare nrcds   address,
                        skip    byte,
                        set     byte,
                        savin   address;

522  2           savin = nrcds;

523  2     cp20: if savin > maxrcd then set = maxrcd;
525  2           else set = savin;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
526  2            call setmulti(set);
527  2            flag = readsq(comptr);                    /* get nrcds units */

528  2       cp21:  if skip = 0 then flag = writesq(.tempfcb);    /* while savin > 0 */
530  2            savin = savin - set;

531  2            if savin = 0 then return;

533  2            if savin > maxrcd then set = maxrcd;
535  2            else set = savin;

536  2            call setmulti(set);
537  2            flag = readsq(comptr);

538  2            go to cp21;

539  2       end copy2;

            /*****************************************************************************/


540  1       reopen$temp: procedure;
541  2            declare i       byte;

542  2            call closer(.tempfcb);
543  2            call clearfcb(.tempfcb);
544  2            call opener(.tempfcb);

545  2            call setmulti(2);

546  2            readflag = readsq(.tempfcb);

547  2       end reopen$temp;


            /*****************************************************************************/


548  1       get$off: procedure(xrecs,index);
549  2            declare index   byte,
                         xrecs   address,
                         i       based xrecs address;
550  2            declare (temp,sum)      address;

551  2       gt0:     temp = offsets(index - 1);
552  2            sum = temp + i;
553  2       gt1:     if sum < temp then call err$print(.err$msg$toobig);

555  2            offsets(index) = sum;

556  2       end get$off;

557  1       zapRSX: procedure;

558  2            declare dRSX    based subptr (16) byte,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        i       byte;

559  2          do i = 0 to 15;
560  3                  dRSX(i) = 0;
561  3          end;

562  2          subptr = subptr + 16;

563  2     end zapRSX;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
      /*************************************************************************/


564  1     addrsx: procedure;
565  2          declare i              byte,
                     prlptr           address,
                     rsxlen           based prlptr address;

566  2          i = 1;
567  2     next$rsx:    fcbp = allfcbs(i);          /* while i <= incount */

568  2          call setmulti(2);              /* get header */
569  2          readflag = readsq(fcbp);
570  2          prlptr = .iobuff(1);           /* get program length */
571  2     ad1: length$rsx(i) = rsxlen;

572  2          call setmulti(1);
573  2          readflag = readsq(fcbp);

574  2          if iobuff(15) <> 0 then iobuff(14) = 0ffh;
576  2          nbank(i) = iobuff(15);         /* only non-banked ? */
577  2          iobuff(10) = 6;
578  2          iobuff(12) = 7;
579  2          iobuff(24) = 0;

580  2          writeflag = writesq(.tempfcb);

581  2          rsxrec = 128;
582  2          call copy(.rsxrec);

583  2     ad2: totbyte = totbyte + rsxrec;

584  2          i = i + 1;

585  2          if i > incount then go to fini;

587  2          call get$off(.rsxrec,i);
588  2          go to next$rsx;

589  2     fini:  end addrsx;


      /*************************************************************************/


590  1     putSCBcode: procedure(ptrfcb);
```

```
591  2          declare (i,j)   byte,
                        ptrfcb  address,
                        fixup   address,
                        fa      based fixup address;

592  2          if not SCB  and not oldSCB then return;

594  2          totbyte = totbyte + 256;        /* rel to 100h */

595  2          call setdma(.SCBbuff);
596  2          call setmulti(2);

597  2          if oldscb then i = SCBbuff(23); /* next open slot */
599  2          else if SCB then do;            /* must initialze buffer with code */

601  3                  do i = 0 to 255;
602  4                          SCBbuff(i) = 0ffh;
603  4                  end;

604  3     ps0:         fixup = .SCBcode(1);
605  3                  fa = fa + totbyte;
606  3                  fixup = .SCBcode(19);
607  3                  fa = fa + totbyte;

608  3     ps1:         call move(23,.SCBcode,.SCBbuff(0));
609  3                  i = 24;
610  3          end;

            ps2:  if nscbs > 0 then do;
613  3             do j = 0 to nscbs-1;
614  4                  SCBbuff(i) = soff(j);
615  4                  SCBbuff(i+2) = sval(j);
616  4                  i = i + 3;
617  4             end;
618  3          end;

619  2          SCBbuff(23) = i;                /* next available scb init */

620  2     ps3: if oldSCB then
621  2                  if ptrfcb = comptr then comfcb(32) = comfcb(32) - 2;

623  2          writeflag = writesq(ptrfcb);
624  2          call setdma(.iobuff);

625  2     end putSCBcode;

           /*****************************************************************************/


626  1     update$head: procedure;
627  2          declare (i,j,K)         byte,
                        (olds,temp)     byte;


628  2      possub: procedure;

629  3          subptr = .iobuff(16);           /* start of RSX info in header */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
630  3          i = 1;                          /* skip old rsx heads */
631  3          do while i <= old;
632  4                 subptr = subptr + 16;
633  4                 i = i + 1;
634  4          end;
635  3     end possub;

           /*************************************************************/


636  2          call possub;                    /* set subptr to end of RSX */
637  2          head$ptr = .iobuff;

638  2          if not COMonly then do;
640  3                 if build then head.progsize = comsize;
642  3     up1:        k = old;

643  3                 do i = 1 to incount;
644  4                       k = k + 1;
645  4                       rsx$sub$head.off = offsets(i);
646  4                       rsx$sub$head.len = length$rsx(i);
647  4                       rsx$sub$head.NONBANK = nbank(i);
648  4                       fcbp = allfcbs(i);
649  4                       do j = 0 to 7;
650  5                             rsx$sub$head.name(j) = gen$fcb(j + 1);
651  5                       end;

652  4                       subptr = subptr + 16;
653  4                 end;
654  3          end;                            /* COMonly... */
655  2          else head.progsize = comsize;

656  2     up2: if LOAD then head.LOADER = 1;
658  2          if SCB or oldSCB then call move(2,.totbyte,.iobuff(4));

660  2          tempfcb(32) = 0;                      /* backup CR to re-write rcd */

661  2          writeflag = writesq(.tempfcb);
662  2          call closer(.tempfcb);

663  2          if not NULL then call deleter;        /* erase old file */
665  2          call renamer;

666  2     end update$head;

           /*************************************************************************/


667  1     tear$down: procedure;

                                                /* remove header from file */
668  2          head$ptr = .iobuff(0);
669  2          comsize = head.progsize/128;

670  2     tri: call copy2(comsize,0);          /* copies com to temp */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
671  2          call closer(comptr);
672  2          call closer(.tempfcb);
                                          /* set up pass if any */
673  2          if len$pass(0) > 0 then call copypass$dma(0);
675  2          call deleter;                  /* delete com file*/
676  2          call renamer;

677  2     end tear$down;


       /*****************************************************************************/

678  1     create2: procedure;


679  2          if not COMonly then do;

681  3               offsets(0) = 256;              /* starting pos in bytes */
682  3     cr4:      call get$off(.comsize,1);
683  3               call addrsx;                   /* copy RSX to temp */
684  3          end;

685  2          call putSCBcode(.tempfcb);

686  2          call reopen$temp;

687  2     cr5: old = 0;
688  2          call update$head;

689  2     end create2;

       /*****************************************************************************/


690  1     create: procedure;
691  2          declare i      byte;

692  2          do i = 0 to 384;                    /* clear the header buffer */
693  3               head$buffer(i) = 0;
694  3          end;
695  2          do i = 0 to incount;                /* clear offsets */
696  3               offsets(i) = 0;
697  3          end;

698  2          head$ptr = .head$buffer;
699  2          head.retinst = ret$inst;
700  2          if not SCB then head.SCBjmp = ret$inst;
702  2          else head.SCBjmp = 0c3h;

703  2          head.nrsx = incount;

704  2          totbyte = 256;
705  2          if NULL then do;
707  3               head$buffer(256) = ret$inst;
708  3               call setmulti(3);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
709   3           end;

710   2     cr1:  call setdma(head$ptr);                /* move dma to header */
711   2           writeflag = writesq(.tempfcb);
712   2           if writeflag > 0 then do;
714   3                 fcbp = .tempfcb;
715   3                 call err$print(.err$msg$write);
716   3           end;

717   2           call setdma(.iobuff);

718   2           if not NULL then do;

720   3                 if readflag <> 1 then do;          /* if size of COM = 1
                                                            then read in setup
                                                            found EOF, no need
                                                            to copy; if flag > 1
                                                            then setup catches */

722   4                       writeflag = writesq(.tempfcb); /* first 2 COM rcds */

723   4                       fcbp = comptr;
724   4                       comsize = 256;
725   4     cr2:              call copy(.comsize);         /* COM->temp */
726   4                 end;
727   3                 else do;
728   4                       call setmulti(1);
729   4                       writeflag = writesq(.tempfcb);
730   4                       comsize = 128;
731   4                 end;
732   3           end;
733   2           else comsize = 128;

734   2           totbyte = totbyte + comsize;

735   2           call create2;

736   2     end create;

      /*****************************************************************************/


737   1     SCBget: procedure(skip);
738   2           declare  skip   byte;
                                          /* where in record units is beginning
                                             of SCB initialization code?
                                             Record numbering is rel to 0 */

739   2                 comsize = shr(SCBpos,7) - 4;
740   2                 call copy2(comsize,skip);  /* do not copy SCB code */
741   2                 totbyte = shl(comsize,7);

742   2                 readflag = readsq(comptr);
743   2                 call move(256,.iobuff,.SCBbuff);

744   2     end SCBget;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
        /*****************************************************************************/

745  1  remover: procedure;
                                        /* remove old RSX in gencommed file */

746  2  getname: procedure(j);

747  3        declare (j,K)   byte;

748  3        do K = 0 to 7;
749  4              new(j).name(K) = rsx$sub$head.name(K);
750  4        end;
751  3  end getname;

752  2        declare (i,j,K,l)       byte,
                 zeroes          based subptr (1) byte,
                 tot             address;

753  2        fcbp = comptr;
754  2  rp1:  subptr = .iobuff(16);              /* prepare to collapse header..
                                                    compute actual lengths,
                                                    & save start bit map */
755  2        nextptr = .iobuff(32);
756  2        do j = 1 to old;
757  3              newlen(j) = rsx$sub$head.len;   /* save len & name */
758  3              call getname(j);
759  3              actlen(j) = next.off - rsx$sub$head.off;
760  3              nbank(j) = rsx$sub$head.NONBANK;

761  3              subptr = nextptr;
762  3              nextptr = nextptr + 16;
763  3        end;
764  2        actlen(old) = 0;

765  2  rp2:  subptr = .iobuff(16);              /* start copying current COM
                                                    file, skipping dup entries*/
766  2        writeflag = writesq(.tempfcb);    /* header */
767  2        tot = shr(head.progsize,7);       /* # 80h units to copy */
768  2        call copy2(tot,0);                /* copies COM to temp */
769  2        tot = tot + 2;

770  2  rp3:  j = 1;                            /* now copy each valid RSX */
771  2        do i = 1 to old;
772  3              comsize = shr(actlen(i),7);     /* convert to 80h units */
773  3              if which(i) = i then do;        /* duplicate */
775  4                    if i <> old then          /* don't skip last */
776  4                      call copy2(comsize,1);
777  4              end;
778  3              else do;                        /* copy RSX & setup new offsets
                                                       lengths */
779  4  rpx:                newoff(j) = shl(tot,7);
780  4                    nbank(j) = nbank(i);
                                                  /* if last RSX then we have no
                                                     way of knowing the actual
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                                        length...so write until EOF,
                                                        else write comsize # rcds */
781  4                          if i = old then call copy(.tot);
783  4                          else do;
784  5                                 tot = tot + comsize;
785  5                                 call copy2(comsize,0);
786  5                          end;

787  4                          newlen(j) = newlen(i);          /* i > j always */
788  4                          do k = 0 to 7;
789  5                                 new(j).name(k) = new(i).name(k);
790  5                          end;
791  4                          j = j + 1;
792  4                    end;
793  3              end;

                                                       /* now rebuild header */
794  2              call reopen$temp;

795  2              j = j - 1;
796  2              subptr = .iobuff(16);
797  2              do i = 1 to j;                    /* j = # good RSX */
798  3                     rsx$sub$head.off = newoff(i);
799  3                     rsx$sub$head.len = newlen(i);
800  3                     rsx$sub$head.NONBANK = nbank(i);
801  3                     nbank(i) = 0;
802  3                     do k = 0 to 7;
803  4                            rsx$sub$head.name(k) = new(i).name(k);
804  4                     end;
805  3                     subptr = subptr + 16;
806  3              end;

807  2              do i = j + 1 to old;              /* clear out header */
808  3                     call zapRSX;
809  3              end;

810  2      rp4:    head.nrsx = j;
811  2              old = j;

812  2              tempfcb(32) = 0;                  /* CR = 0 */
813  2              flag = writesq(.tempfcb);

814  2              call closer(.tempfcb);            /* close and rename */
815  2              call deleter;                     /* delete com file */
816  2              call renamer;

817  2              call clearfcb(comptr);
818  2              call clearfcb(.tempfcb);
819  2              call maker(.tempfcb);
820  2      rp9:    call opener(comptr);              /* prepare return to concat */
821  2      rp7:    readflag = readsq(comptr);

822  2      end remover;

        /*****************************************************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
823  1     dup$RSX: procedure byte;
                                        /* check for duplications in header and
                                           input.  Remove old entry if found,
                                           or if all are duplicated then strip
                                           everything off. */

824  2           declare (i,j,k,l)      byte,
                     temp           address;

825  2           subptr = .iobuff(16);
826  2           deletes = 0;

827  2           do i = 1 to old;
828  3                 which(i) = 0;

829  3                 do j = 1 to incount;          /* compare names */
830  4                       fcbp = allfcbs(j);
831  4                       do k = 0 to 7;
832  5                             if rsx$sub$head.name(k) <> gen$fcb(k+1)
                                         then go to dp1;
834  5                       end;
                                                    /* duplicate RSX's */
835  4                       which(i) = i;
836  4                       deletes = deletes + 1;

837  4                       call e$print1(.err$msg$dup2);
838  4                       call e$print2;

839  4                       go to dp2;            /* no need to scan rest of
                                                     input names- checked input
                                                     for dups already */
840  4     dp1:              end;
841  3     dp2:              subptr = subptr + 16;
842  3           end;

843  2           if deletes = 0 then return(false);
845  2     dp4:  if deletes >= old then do;          /* replace all ? */
847  3                 subptr = .iobuff(16);
848  3                 do i = 1 to old;
849  4                       call zapRSX;
850  4                 end;

851  3                 temp = head.progsize;       /* get size of COM in recds */

852  3                 if oldSCB then do;
854  4                       call SCBget(1);
855  4                       comfcb(32) = 0;
856  4                       call setmulti(2);
857  4                       readflag = readsq(comptr);
858  4                 end;

859  3                 comsize = shr(temp,7);
860  3                 writeflag = writesq(.tempfcb);  /* copy header to temp */
861  3                 call copy2(comsize,0);          /* copy COM file */

862  3                 comsize = temp;               /* back to byte count */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
863  3                  call create2;

864  3                  return(true);
865  3            end;

866  2            call remover;                          /* selective replace */

867  2            return(false);                         /* return and add new RSX */

868  2      end dup$RSX;


       /*****************************************************************************/


869  1      concat: procedure;
                                                  /* add new, replace old */

870  2            declare i      byte;

871  2            head$ptr = .iobuff;
872  2            if (old := head.nrsx) <> 0 then do;
874  3      yy:         if dup$RSX then return;          /* true  : did a create
                                                           false : add new RSX,
                                                                   might have collapsed
                                                                   old header...*/

876  3            end;

877  2            head.nrsx = head.nrsx + incount;
878  2            fcbp = comptr;

879  2      cc1:  if head.nrsx > 15 then
880  2                  call err$print(.err$msg$rsx$slot);

881  2            flag = writesq(.tempfcb);        /* write header */

882  2            if oldSCB then call SCBget(0);
884  2            else do;                         /* no SCB...copy to EOF */
885  3                  comsize = 256;
886  3                  call copy(.comsize);
887  3            end;

                                                  /* comsize = size of file in bytes
                                                     +1 = offset of first new RSX */
888  2            offsets(0) = 0;
889  2            call getoff(.comsize,1);

890  2            totbyte = comsize;

891  2            call closer(fcbp);               /*close old file */

892  2            call addrsx;

893  2            call putSCBcode(.tempfcb);

894  2            call reopen$temp;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
895  2          call update$head;

896  2     end concat;

           /*****************************************************************************/

897  1     setSCB: procedure;

                                    /* read in gencommed file and set scb values
                                       from command line */

898  2          head$ptr = .iobuff;

899  2          fcbp = comptr;
900  2          totbyte = 2;

901  2          if LOAD then do;               /* write out loader flag */
903  3               if oldSCB or not SCB then do;
905  4                    iobuff(13) = 1;
906  4                    comfcb(32) = 0;
907  4                    writeflag = writesq(.comfcb);
908  4                    if writeflag <> 0 then call err$print(.err$msg$write);
910  4                    totbyte = 0;
911  4               end;
912  3          end;

913  2          if SCB then do;
915  3               if oldSCB then call SCBget(1);
917  3               else do;
918  4                    if readflag <> 1 then do;          /* 1 rcd com file */
920  5                         call setmulti(32);
921  5                         call mread(comptr);
922  5                         do while readflag <> 1;
923  6                              totbyte = totbyte + nrecs;
924  6                              call mread(comptr);
925  6                         end;
926  5                    end;

927  4                         totbyte = totbyte + nrecs;
928  4                         totbyte= shl(totbyte,7);       /* change to bytes */
929  4               end;

930  3               call putSCBcode(comptr);

931  3               if not oldSCB then do;                 /* must update header
                                                              for new SCB's */
933  4                    call closer(comptr);
934  4                    call setmulti(1);
935  4                    call clearfcb(comptr);
936  4                    call opener(comptr);
937  4                    readflag = readsq(comptr);
938  4                    call move(2,.totbyte,.iobuff(4));
939  4                    if LOAD then iobuff(13) = 1;
941  4                    iobuff(3) = ret$inst;
942  4                    comfcb(32) = 0;
943  4                    writeflag = writesq(.comfcb);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #: _____

```
944  4                          if writeflag <> 0 then call err$print(.err$msg$write);
946  4                    end;
947  3             end;

948  2             call closer(comptr);

949  2     end setSCB;

           /*************************************************************************/

950  1     setuper: procedure;

                                        /*      1. get each file (process passwords)
                                                2. check for proper type
                                                3. check for duplicate RSX on input
                                                4. open files and make temp
                                        */

951  2             declare (i,j,K,l)       byte;

952  2     init:   procedure;

953  3             fcbp,allfcbs(i) = .fcbs(i).file(0);
954  3             do j = 0 to 32;
955  4                   fcbs(i).file(j) = 0;
956  4             end;
957  3     end init;

958  2     RSX$errprint: procedure;


959  3             call e$print1(.('This file was not used.',0));
960  3             call e$print2;
961  3             call crlf;

962  3             which(deletes) = i;
963  3             deletes = deletes + 1;

964  3     end RSX$errprint;

965  2     fill$type: procedure(typea);
966  3             declare typea   address,
                            type    based typea (1) byte;

967  3             K = 0;
968  3             do l = 9 to 11;
969  4                   gen$fcb(l) = type(K);
970  4                   K = K + 1;
971  4             end;

972  3     end fill$type;


973  2     checktype: procedure(typea) byte;
974  3             declare typea   address,
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        type    based typea (1) byte;

 975  3             if gen$fcb(9) = BLANK then              /* any type ? */
 976  3                     call fill$type(typea);

 977  3             else do;                                /* check input type */
 978  4                     K = 0;
 979  4                     do l = 9 to 11;
 980  5                             if gen$fcb(l) <> type(K) then return(false);
 982  5                             K = k + 1;
 983  5                     end;
 984  4             end;

 985  3             return(true);

 986  3     end checktype;



 987  2             buf$ptr = .buff(1);                     /* get files */
 988  2             i = 0;
 989  2             do while buf$ptr <> 0;
 990  3                     call init;
 991  3                     call parser(fcbp);

 992  3                     if optmark = '[' then go to sb1;/* no more names, options */

                                                            /* any PASSWORDS !!!! */
 994  3                     K = gen$fcb(26);                /* length of password */
 995  3                     if k > 0 then do;
 997  4                             l = 16;                 /* start of password */
 998  4                             do j = 0 to k - 1;
 999  5                                     files(i).pass(j) = gen$fcb(l);
1000  5                                     l = l + 1;
1001  5                             end;
1002  4                             len$pass(i) = K;
1003  4                     end;
1004  3                     i = i + 1;
1005  3             end;

1006  2     sb1:    incount = i - 1;

1007  2             if optmark = '[' then do;
1009  3                     incount = i;
1010  3                     call getoption;
1011  3             end;

1012  2             comptr = allfcbs(0);
                                                            /* check COM */
1013  2     sb2:            fcbp = comptr;
1014  2                     if not checktype(.comtype) then do;     /* bad input */
1016  3                       if not NULL then do;
1018  4                             call print(.err$msg$first);
1019  4                             call terminate;
1020  4                       end;
1021  3                     end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1022  2                   if len$pass(0) > 0 then call copypass$dma(0);
1024  2                   if open(fcbp) > 3 then do;            /* something awry */
1026  3                     if not NULL then do;
1028  4                           call err$print(.err$notfnd);
1029  4                           call e$print1(.err$msg$first);
1030  4                           call terminate;
1031  4                     end;
1032  3                   end;
                          else
1033  2                   if NULL then
1034  2                   if (comfcb(8) and 80h) <> 80h then
1035  2                           call err$print(.err$NULL); /* NULL and COM file*/

                          if NULL then do;
1038  3       sb3:        i = (incount != incount + 1);   /* move fcbs up */
1039  3                   allfcbs(i) = .fcbs(i);
1040  3                   do j = 0 to incount - 1;
1041  4                           do k = 0 to 32;
1042  5                                   fcbs(i).file(k) = fcbs(i-1).file(k);
1043  5                           end;
1044  4                           i = i - 1;
1045  4                   end;
                                                          /* dummy COM name = 1st RSX */
1046  3                   call fill$type(.comtype);
1047  3                   fcbp = allfcbs(1);              /* restore type to RSX */
1048  3                   call fill$type(.rsxtype);
1049  3             end;

1050  2       sb4:  if incount > 0 then do;
1052  3                   deletes = 0;                    /* now check RSX's */
1053  3                   do i = 1 to incount;
1054  4                           fcbp = allfcbs(i);      /* point to RSX fcb */

1055  4                           if not checktype(.rsxtype) then do;
1057  5                                   call e$print1(.err$msg$rsxval);
1058  5                                   call RSX$errprint;
1059  5                           end;

1060  4                           else do;                /* try to open file */
1061  5                                   if len$pass(i) > 0 then
1062  5                                      call copypass$dma(i);

1063  5                                   flag = open(fcbp);
1064  5                                   if flag > 3 then do;
1066  6                                          call e$print1(.err$notfnd);
1067  6                                          call RSX$errprint;
1068  6                                   end;
                                            else          /* Duplicate input RSX ? */
1069  5                                        do j = i+1 to incount;
1070  6                                          test$ptr = allfcbs(j);
1071  6                                          do l = 1 to 8;
1072  7                                             if genfcb(l) <> testfcb(l)
                                                        then go to sb5;
1074  7                                          end;
1075  6                                          call e$print1(.err$msg$dup1);
1076  6                                          call RSX$errprint;
1077  6       sb5:                             end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1078  5                         end;
1079  4                   end;                          /* ends i = incount...*/

                                                        /* have any RSX's left? */
1080  3                   if deletes >= incount then do;
1082  4                      call print(.err$msg$no$rsx);
1083  4                      call terminate;
1084  4                   end;

1085  3                   i = 0;
1086  3     sb6:          do while i < deletes;   /* collapse allfcbs */
1087  4                         j = which(i);
1088  4                         incount = incount - 1;

1089  4                         do l = j to incount;
1090  5                               allfcbs(l) = allfcbs(l + 1);
1091  5                         end;

1092  4                         i = i + 1;
1093  4                   end;

1094  3                   rsx = true;
1095  3             end;                          /* if incount> 0...*/

1096  2     sb7:
                    call setdma(.iobuff);
1097  2             call setmulti(2);                      /* read header if any */

1098  2             if not NULL then do;
1100  3                   fcbp = comptr;
1101  3                   call mread(comptr);
1102  3                   if readflag > 1 then call err$print(.err$msg$read);

                                                        /* is this already gencommed*/
1104  3     sb8:          if iobuff(0) = ret$inst then do;
                                                        /* first byte = return */
1106  4                         if rsx then replace = true;
1108  4                         else do;
1109  5                               if SCB or LOAD then punchSCB = true;
1111  5                               else revert = true;
1112  5                         end;

                                                        /* do we need to move old SCB
                                                           initialization code ? */
1113  4                         if iobuff(3) <> 0c9h then do;
1115  5                               oldSCB = true;
1116  5                               call move(2,.iobuff(4),.SCBpos);
1117  5                         end;
1118  4                   end;
1119  3                   else do;
1120  4                         if rsx then build = true;
1122  4                         else if SCB or LOAD then COMonly = true;
1124  4                         else call err$print(.errSTRIP);
1125  4                   end;
1126  3             end;
1127  2             else build = true;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1128  2     sb9:    if not punchSCB then do;
1130  3                     call clearfcb(.tempfcb);
1131  3                     flag = delete(.tempfcb);
1132  3                     tempfcb(0) = comfcb(0);        /* init temp drive */
1133  3     sb0:            call maker(.tempfcb);
1134  3             end;

1135  2     end setuper;



              /*                    MAIN PROGRAM                    */


1136  1     plm:

                      testvers = get$version;
1137  1             if high(testvers) = 1 then go to err$vers;
1139  1             if low(testvers) < 30h then go to err$vers;

1141  1             call return$errors(254);

1142  1             call setuper;

1143  1             if revert then call tear$down;
                      else
1145  1             if build then call create;
                      else
1147  1             if punchSCB then call setscb;
1149  1             else if COMonly then call create;
1151  1             else call concat;

1152  1             call closeall;

1153  1             call print(.('GENCOM completed.',0));
1154  1             call terminate;

1155  1     err$vers:
                      call print(.ERRORM);
1156  1             call printx(.('Requires CP/M 3 or higher.',0));
1157  1             call terminate;


1158  1     end gencomer;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
MODULE INFORMATION:

    CODE AREA SIZE     = 20A8H   8360D
    VARIABLE AREA SIZE = 17CDH   6093D
    MAXIMUM STACK SIZE = 0018H     24D
    1999 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 GENCOM
0000 MCD80A
0004 BDISK    0080 BUFF      0080 BUFFA     0050 CMDRV     0000 CPU
007C CR             006D DOLLA     005C FCB       006C FCB16     005C FCBA
005C IFCB     005C IFCBA     0003 IOBYTE    0053 LEN0      0056 LEN1
0006 MAXB     0006 MEMSIZ    0005 MON1      0005 MON2      0005 MON2A
0005 MON3     0000 OFFSET    006E PARMA     0051 PASS0     0054 PASS1
007F R0             007D RR        007D RRECA     007F RRECQ     005C SFCB
0080 TBUFF
0000 PARSE
02C9 DBLNK1   02BD DEBLNK    027B DELIM     02B6 DELIM1    02B9 DELIM2
0273 PADFLD   0211 PARS81    0212 PARS82    0180 PARSE     01A1 PARSE1
01CF PARSE2   01D2 PARSE3    01E4 PARSE4    01E8 PARSE5    01EA PARSE6
01FE PARSE7   0202 PARSE8    0223 PARSE9    024C PWFLD     025F PWFLD1
023E SETFD1   0240 SETFD2    0244 SETFD3    0229 SETFLD
0000 GENCOMER
3C3B MEMORY   0627 PLM       02CD COPYRIGHT 02F3 VERSION   02FB RESETMASK
06A4 PRINTCHAR 246E CHAR     06B4 PRINTB    06BA PRINTX    246F A
06DB CHECKCONSTAT            06E4 CRLF      0706 PRINT     2471 A
0718 GETVERSION      0721 CONIN     072A OPEN      2473 FCB       073A CLOSE
2475 FCB      074A MAKE      2477 FCB       2479 IOFLAG    247B NRECS
075A MREAD    247C FCB       0778 SETMULTI  247E NSECTS    078B READSQ
247F FCB      079B WRITESQ   2481 FCB       07AB RENAME    2483 FCB
07BB DELETE   2485 FCB       07CB SETDMA    2487 DMA
07DB RETURNERRORS            2489 MODE      07EB TERMINATE 031B OPTIONS
032B OFFOPT   032F ENDLIST   0330 ENDOFSTRING              0331 DELIMITERS
034C SPACE    034D COMMA     034E LPAREN    248A OPTMAP    24A1 J
24A2 BUFPTR   24A4 OPTINDEX  24A5 ENDBUF    24A6 DELIMITER 24A7 PUNCHSCB
24A8 COMONLY  24A9 REVERT    24AA BUILD     24AB REPLACE   24AC EMPTY
24AD HEX .    24AE OLDSCB    24AF INCOUNT   034F RETINST   0350 BLANK
24B0 READFLAG 24B1 WRITEFLAG 24B2 FLAG      24B3 RSX       24B4 OLD
24B5 FILL     0351 MAXRCD    24B6 DELETES   24B7 WHICH     24C6 COMOFF
24C8 COMSIZE  24CA TOTBYTE   24CC RSXREC    24CE OLDRSX    24D0 OFFSETS
24EE LENGTHRSX 250C TESTVERS 0352 COMTYPE   0355 HEXTYPE   0358 RSXTYPE
250E TEMPFCB  252F ERRFCB    253D FILES     25BD LENPASS   25CD PARSESTRUC
25D1 NULL     25D2 LOAD      25D3 SCB       25D4 FCBS      27E4 TESTPTR
27E6 ALLFCBS  2806 FCBP      2808 COMPTR    280A HEADPTR   280C SUBPTR
280E HEADBUFFER      298E IOBUFF    398E NEXTPTR   3990 NBANK     39A0 NEWOFF
39C0 NEWLEN   39E0 ACTLEN    39FE NEW       3A76 SOFF      3A8A SVAL
3A9E NSCBS    3A9F SCBBUFF   035B SCBCODE   3B9F SCBPOS    0372 ERRORM
037A FILEM    0381 ERRNOTFND 0391 ERRMSGMAKE 03A5 ERRMSGPARSE
03B8 ERRMSGFIRST             03E1 ERRMSGDUP1 03F8 ERRMSGDUP2
0427 ERRMSGRSXVAL            0439 ERRMSGNORSX               0457 ERRMSGCOPY
0466 ERRMSGRSXSLOT           0490 ERRMSGREAD 049B ERRMSGWRITE
04A7 ERRMSGTOOBIG            04C4 ERRNULL   04E4 ERRSTRIP  0500 ERRIFCB
050D ERRMEDIA 0524 ERRDIO    0534 ERRDRIVE  0549 ERRSCBOFF
055C ERRSCBCLOSE             0577 ERRSCBNOVAL               058A ERRSCBPAR
05A4 ERRUNRECOPT             05B9 ERRNOTSCB 07F7 CLOSEALL  3BA1 I
0833 GETERRFCB 3BA2 I        3BA3 J         089E GE1       08ED EPRINT1
3BA4 MESSAGE  0905 EPRINT2   0915 ERRPRINT  3BA6 MESSAGE   092D BDOSERR
3BA8 LFLAG    3BA9 HFLAG     0979 SEPARATOR 3BAA CHARACTER 3BAB K
0982 LOOP     09B1 OPTSCANNER 3BAC LISTPTR  3BAE OFFPTR    3BB0 IDXPTR
3BB2 I        3BB3 J         3BB4 WRDPOS    3BB5 CHARACTER
3BB6 LETTERINWORD            3BB7 FOUNDFIRST 3BB8 START    3BB9 SAVEINDEX
3BBA LENNEW   3BBB LENFOUND  3BBC VALID     0AD8 CHECKINLIST
3BBD I        0B2E SETUP     0B4C TESTLETTER 0B8C SKIP      0C03 EATBLANKS
0A11 EXIT1    0A21 GOOD      0A36 NEXTOPT   0AA7 FINISHED  0C1F ERRORPRT
3BBE I        3BBF T         0C51 EPRINT3   3BC1 MESSAGE   0C69 ASCHEX
3BC3 AHBYTE   3BC5 ALBYTE    0C97 CONV      3BC7 ABYTE     0CB8 VALOFF
3BC9 HIGH     3BCA LOW       3BCB ACHAR     0CE2 GETOPTION 3BCD BUFPTR1
3BCF INDEX    3BD0 ZERO      0DA7 GETSCBVAL 0E32 CHECKVAL  0E59 CV0
0E61 GETSCBOFF 0EEA GS1      0CF8 GT00      0D88 GT01      0F0F OPENER
3BD1 FCB      0F31 CLOSER    3BD3 FCB       0F53 MAKER     3BD5 FCB
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0F7A DELETER   0FA0 PARSER    3BD7 FCBPTR    0FB8 PA1       0FC1 PA2
0FDB COPYPASSDMA              3BD9 INDEX     3BDA I         1016 RENAMER
3BDB I         3BDC J         3BDD RENBUF    108D RE1       10A6 CLEARFCB
3BFD FCB       3BFF I         10CE COPY      3C00 RECSIZE   3C02 I
3C03 FLAG      10E3 CO2       1169 COPY2     3C05 NRCDS     3C07 SKIP
3C08 SET       3C09 SAVIN     1177 CP20      11A6 CP21      1202 REOPENTEMP
3C0B I         1223 GETOFF    3C0C XRECS     3C0E INDEX     3C0F TEMP
3C11 SUM       122B GTO       124C GT1       1272 ZAPRSX    3C13 I
129E ADDRSX    3C14 I         3C15 PRLPTR    12A3 NEXTRSX   12CA AD1
132E AD2       1354 FINI      1355 PUTSCBCODE 3C17 PTRFCB   3C19 I
3C1A J         3C1B FIXUP     13B8 PS0       13DC PS1       13F1 PS2
144E PS3       1483 UPDATEHEAD 3C1D I        3C1E J         3C1F K
3C20 OLDS      3C21 TEMP      15C1 POSSUB    14A8 UP1       1573 UP2
15EB TEARDOWN  1602 TR1       162F CREATE2   163D CR4       1651 CR5
165A CREATE    3C22 I         16F8 CR1       1749 CR2       177E SCBGET
3C23 SKIP      17CB REMOVER   1ACB GETNAME   3C24 J         3C25 K
3C26 I         3C27 J         3C28 K         3C29 L         3C2A TOT
17D1 RP1       1863 RP2       1890 RP3       18DB RPX       1A78 RP4
1AB7 RP9       1ABF RP7       1B0E DUPRSX    3C2C I         3C2D J
3C2E K         3C2F L         3C30 TEMP      1B80 DP1       1B8A DP2
1BD9 DP4       1C65 CONCAT    3C32 I         1C7B YY        1C99 CC1
1CFC SETSCB    1E3A SETUPER   3C33 I         3C34 J         3C35 K
3C36 L         225F INIT      22B2 RSXERRPRINT              22D2 FILLTYPE
3C37 TYPEA     2313 CHECKTYPE 3C39 TYPEA     1EDB SB1       1EFA SB2
1F76 SB3       201D SB4       20F9 SB5       2125 SB6       2189 SB7
21B9 SB8       223A SB9       2258 SB0       0693 ERRVERS
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #: _____

```
0000 GENCOM#
0000 MCD80A#
0000 PARSE#
0000 GENCOMER#
06A4 18          06A8 20          06B3 21          06B4 22          06B4 23
06B9 24          06BA 25          06C0 28          06C9 29          06D0 30
06D7 31          06DA 32          06DB 33          06DB 34          06E4 35
06E4 36          06E4 37          06E9 38          06EE 39          06F5 41
06FD 42          0705 43          0705 44          0706 45          070C 47
070F 48          0717 49          0718 50          0718 51          0721 52
0721 53          0721 54          072A 55          072A 56          0730 58
073A 59          073A 60          0740 62          074A 63          074A 64
0750 66          075A 67          075A 69          0760 71          076C 72
0773 73          0777 74          0778 75          077C 77          078A 78
078B 79          0791 81          079B 82          079B 83          07A1 85
07AB 86          07AB 87          07B1 89          07BB 90          07BB 91
07C1 93          07CB 94          07CB 95          07D1 97          07DA 98
07DB 99          07DF 101         07EA 102         07EB 103         07EB 104
07EE 105    07F6 106    07F7 116    07F7 118    0806 119
0819 120    0820 121    0829 122    0832 123    0833 124
0833 126    0841 127    084C 128    0853 129    0858 130
085D 131    0860 132    087E 133    0895 134    0899 135
089B 136    089E 137    08A9 138    08AE 139    08CC 140
08D0 141    08E5 142    08E9 143    08EC 144    08ED 145
08F3 147    08F6 148    08FC 149    0904 150    0905 151
0905 152    090B 153    0911 154    0914 155    0915 156
091B 158    0923 159    0926 160    0929 161    092C 162
092D 163    092D 165    0934 166    0938 167    0940 168
0946 169    094E 170    0954 171    095C 173    0964 174
096A 175    0972 176    0978 177    0978 178    0979 179
097D 181    0982 182    0992 183    0995 184    09A5 185
09A9 186    09AD 187    09B0 188    09B1 189    0AD8 192
0AD8 194    0ADE 195    0AE9 196    0AFA 197    0AFE 198
0B05 199    0B16 200    0B17 201    0B25 202    0B28 203
0B2D 204    0B2E 205    0B2E 206    0B2E 207    0B35 208
0B38 209    0B3E 210    0B43 211    0B4B 212    0B4C 213
0B4C 214    0B50 215    0B5E 216    0B66 217    0B77 218
0B7C 219    0B80 220    0B8B 221    0B8C 222    0B8C 223
0B9A 224    0BA4 225    0BD4 226    0BD8 227    0BE3 228
0BED 229    0BF0 230    0BF6 231    0C02 232    0C03 233
0C03 234    0C03 236    0C14 237    0C1B 238    0C1E 239
09C2 240    09C5 241    09CA 242    09CD 243    09D6 244
09DD 245    09E7 246    09EC 247    09F4 248    09F7 249
09FE 250    0A01 251    0A0B 252    0A0E 253    0A11 254
0A14 255    0A17 256    0A1D 257    0A20 258    0A21 259
0A27 260    0A2A 261    0A31 262    0A36 263    0A3D 264
0A40 265    0A49 266    0A4C 267    0A68 268    0A70 270
0A75 271    0A84 272    0A87 273    0A8E 274    0A91 275
0A98 276    0A9B 277    0A9E 278    0AA3 279    0AA6 280
0AA7 281    0AAE 282    0ABC 283    0ABF 284    0AC7 285
0AD1 286    0AD7 287    0C1F 288    0C1F 290    0C2C 291
0C3B 292    0C42 293    0C49 294    0C50 295    0C51 296
0C57 298    0C5D 299    0C65 300    0C68 301    0C69 302
0C97 304    0C9D 306    0CA6 307    0CB0 308    0CB7 309
0C73 310    0C7B 311    0C83 312    0C8C 313    0C97 314
0CB8 315    0CC5 317    0CDB 318    0CE1 319    0CE2 320
0DA7 322    0DA7 323    0DAE 324    0DBD 326    0DD3 327
0DDB 328    0DDE 330    0DF2 331    0DF8 332    0DF8 333
0DFC 334    0E0B 335    0E11 336    0E18 337    0E22 338
0E2A 339    0E31 340    0E32 341    0E32 342    0E3C 343
0E46 344    0E49 345    0E53 346    0E59 347    0E60 348
0E61 349    0E61 350    0E72 352    0E79 353    0E85 354
0E8C 355    0E96 356    0EA0 357    0EA3 358    0EAD 359
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

0EB0 360 0EBC 361 0EC8 362 0EDE 363 0EE6 364
0EE9 365 0EEA 366 0EFE 367 0F05 368 0F08 369
0F0E 370 0CE2 371 0CE7 372 0CEC 373 0CF1 374
0CF8 375 0D05 376 0D0D 378 0D13 379 0D19 380
0D1F 381 0D22 382 0D22 383 0D2A 384 0D32 385
0D3A 386 0D3F 387 0D47 388 0D48 389 0D50 390
0D51 391 0D59 393 0D61 395 0D67 396 0D6D 397
0D7A 398 0D7D 399 0D7D 400 0D80 401 0D83 402
0D88 403 0D88 404 0D90 405 0D91 406 0D99 407
0D9A 408 0DA2 409 0DA3 410 0DA6 411 0F0F 412
0F15 414 0F24 416 0F2A 417 0F30 418 0F30 419
0F31 420 0F37 422 0F46 424 0F4C 425 0F52 426
0F52 427 0F53 428 0F59 430 0F64 431 0F6D 433
0F73 434 0F79 435 0F79 436 0F7A 437 0F7A 438
0F89 439 0F8A 440 0F99 442 0F9F 443 0F9F 444
0FA0 445 0FA6 447 0FAC 448 0FB2 449 0FB8 450
0FC1 451 0FCE 453 0FD4 454 0FDA 455 0FDA 456
0FDB 457 0FDF 459 0FED 460 100E 461 1015 462
1016 463 1016 465 1024 466 102C 467 1037 468
103F 469 1046 470 1054 471 105C 472 1072 473
1086 474 108D 475 1096 476 109F 478 10A5 479
10A5 480 10A6 481 10AC 483 10BA 484 10C6 485
10CD 486 10CE 487 10D4 491 10DB 492 10E3 493
10EB 495 10F3 497 10FB 498 10FC 499 10FF 500
1102 501 1102 502 1108 503 1110 505 1117 506
111D 507 111D 508 1126 509 112E 510 113B 511
113F 512 1142 513 114E 514 1154 515 115C 516
115D 517 1165 518 1168 519 1169 520 1171 522
1177 523 1183 524 118C 525 1194 526 119B 527
11A6 528 11AE 529 11B7 530 11C5 531 11CE 532
11CF 533 11DB 534 11E4 535 11EC 536 11F3 537
11FE 538 1201 539 1202 540 1202 542 1208 543
120E 544 1214 545 1219 546 1222 547 1223 548
122B 551 123E 552 124C 553 1258 554 125E 555
1271 556 1272 557 1272 559 1280 560 128C 561
1293 562 129D 563 129E 564 129E 566 12A3 567
12B4 568 12B9 569 12C4 570 12CA 571 12E0 572
12E5 573 12F0 574 12F8 575 12FD 576 130A 577
130F 578 1314 579 1319 580 1322 581 1328 582
132E 583 1339 584 133D 585 1344 586 1347 587
1351 588 1354 589 1355 590 135B 592 136B 593
136C 594 1376 595 137C 596 1381 597 1388 598
1391 599 1398 601 13A6 602 13B1 603 13B8 604
13BE 605 13CA 606 13D0 607 13DC 608 13EC 609
13F1 610 13F1 611 13FA 613 140A 614 1420 615
1436 616 143E 617 1448 618 1448 619 144E 620
1455 621 1462 622 1471 623 147C 624 1482 625
1483 626 15C1 628 15C1 629 15C7 630 15CC 631
15D6 632 15E0 633 15E7 634 15EA 635 1483 636
1486 637 148C 638 1494 640 149B 641 14A8 642
14AE 643 14BD 644 14C4 645 14DA 646 14F2 647
1506 648 1517 649 1525 650 1545 651 154F 652
1559 653 1563 654 1566 655 1573 656 157A 657
1583 658 158E 659 159E 660 15A3 661 15AC 662
15B2 663 15BA 664 15BD 665 15C0 666 15EB 667
15EB 668 15F1 669 1602 670 160C 671 1614 672
161A 673 1623 674 1628 675 162B 676 162E 677
162F 678 162F 679 1637 681 163D 682 1645 683
1648 684 1648 685 164E 686 1651 687 1656 688
1659 689 165A 690 165A 692 166B 693 1676 694
1680 695 168F 696 169F 697 16A9 698 16AF 699
16B6 700 16BE 701 16CC 702 16D5 703 16E0 704
16E6 705 16ED 707 16F3 708 16F8 709 16F8 710
1700 711 1709 712 1712 714 1718 715 171E 716
171E 717 1724 718 172C 720 1734 722 173D 723
1743 724 1749 725 174F 726 1752 728 1757 729

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

1760 730	1766 731	1766 732	1769 733	176F 734
177A 735	177D 736	177E 737	17B2 739	1793 740
179F 741	17AC 742	17B7 743	17CA 744	17CB 745
1ACB 746	1ACF 748	1ADD 749	1B03 750	1B0D 751
17CB 753	17D1 754	17D7 755	17DD 756	17EC 757
1804 758	180B 759	1825 760	1839 761	183F 762
1849 763	1853 764	1863 765	1869 766	1872 767
187E 768	1888 769	1890 770	1895 771	18A2 772
18B4 773	18C4 775	18CE 776	18D8 777	18DB 779
18F2 780	1905 781	190F 782	1918 784	1923 785
192D 786	192D 787	1947 788	1955 789	1981 790
198B 791	1992 792	1992 793	199C 794	199F 795
19A6 796	19AC 797	19BB 798	19D1 799	19E9 800
19FD 801	1A08 802	1A16 803	1A3C 804	1A46 805
1A50 806	1A5A 807	1A6B 808	1A6E 809	1A78 810
1A83 811	1A89 812	1A8E 813	1A97 814	1A9D 815
1AA0 816	1AA3 817	1AAB 818	1AB1 819	1AB7 820
1ABF 821	1ACA 822	1B0E 823	1B0E 825	1B14 826
1B19 827	1B28 828	1B33 829	1B42 830	1B53 831
1B61 832	1B84 833	1B87 834	1B91 835	1B9E 836
1BA4 837	1BAA 838	1BAD 839	1BB0 840	1BBA 841
1BC4 842	1BCE 843	1BD6 844	1BD9 845	1BE3 847
1BE9 848	1BF8 849	1BFB 850	1C05 851	1C10 852
1C17 854	1C1C 855	1C25 856	1C2A 857	1C35 858
1C35 859	1C40 860	1C49 861	1C53 862	1C59 863
1C5C 864	1C5F 865	1C5F 866	1C62 867	1C65 868
1C65 869	1C65 871	1C6B 872	1C7B 874	1C82 875
1C83 876	1C83 877	1C93 878	1C99 879	1CA6 880
1CAC 881	1CB5 882	1CBC 883	1CC4 885	1CCA 886
1CD0 887	1CD0 888	1CD6 889	1CDE 890	1CE4 891
1CEC 892	1CEF 893	1CF5 894	1CF8 895	1CFB 896
1CFC 897	1CFC 898	1D02 899	1D08 900	1D0E 901
1D15 903	1D21 905	1D26 906	1D2F 907	1D3A 908
1D42 909	1D48 910	1D4E 911	1D4E 912	1D4E 913
1D55 915	1D5C 916	1D64 918	1D6C 920	1D71 921
1D79 922	1D81 923	1D8F 924	1D97 925	1D9A 926
1D9A 927	1DA8 928	1DB5 929	1DB5 930	1DBD 931
1DC5 933	1DCD 934	1DD2 935	1DDA 936	1DE2 937
1DED 938	1DFD 939	1E04 940	1E09 941	1E0F 942
1E18 943	1E23 944	1E2B 945	1E31 946	1E31 947
1E31 948	1E39 949	1E3A 950	225F 952	225F 953
2280 954	228E 955	22A7 956	22B1 957	22B2 958
22B2 959	22B8 960	22BB 961	22BE 962	22CB 963
22D1 964	22D2 965	22D8 967	22DD 968	22E9 969
2301 970	2308 971	2312 972	2313 973	2319 975
2327 976	2332 978	2337 979	2343 980	235E 981
2361 982	2368 983	2372 984	2372 985	2375 986
1E3A 987	1E40 988	1E45 989	1E51 990	1E54 991
1E5C 992	1E65 993	1E68 994	1E73 995	1E7C 997
1E81 998	1E91 999	1EB3 1000	1EBA 1001	1EC4 1002
1ED1 1003	1ED1 1004	1ED8 1005	1EDB 1006	1EE2 1007
1EEB 1009	1EF1 1010	1EF4 1011	1EF4 1012	1EFA 1013
1F00 1014	1F0B 1016	1F13 1018	1F19 1019	1F1C 1020
1F1C 1021	1F1C 1022	1F25 1023	1F2A 1024	1F39 1026
1F41 1028	1F47 1029	1F4D 1030	1F50 1031	1F50 1032
1F53 1033	1F5A 1034	1F69 1035	1F6F 1036	1F76 1038
1F80 1039	1F9E 1040	1FAE 1041	1FBC 1042	1FF0 1043
1FFA 1044	2001 1045	200B 1046	2011 1047	2017 1048
201D 1049	201D 1050	2026 1052	202B 1053	203A 1054
204B 1055	2056 1057	205C 1058	205F 1059	2062 1061
2071 1062	2078 1063	2083 1064	208C 1066	2092 1067
2095 1068	2098 1069	20A9 1070	20BA 1071	20C8 1072
20E3 1073	20E6 1074	20F0 1075	20F6 1076	20F9 1077
2103 1078	2103 1079	210D 1080	2117 1082	211D 1083
2120 1084	2120 1085	2125 1086	212F 1087	213C 1088
2143 1089	2153 1090	2170 1091	217A 1092	2181 1093

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
2184 1094   2189 1095   2189 1096   218F 1097   2194 1098
219C 1100   21A2 1101   21AA 1102   21B3 1103   21B9 1104
21C3 1106   21CA 1107   21D2 1109   21DD 1110   21E5 1111
21EA 1112   21EA 1113   21F2 1115   21F7 1116   2207 1117
2207 1118   220A 1120   2211 1121   2219 1122   2224 1123
222C 1124   2232 1125   2232 1126   2235 1127   223A 1128
2242 1130   2248 1131   2251 1132   2258 1133   225E 1134
225E 1135   0624 1136   0630 1137   0639 1138   063C 1139
0645 1140   0648 1141   064D 1142   0650 1143   0657 1144
065D 1145   0664 1146   066A 1147   0671 1148   0677 1149
067E 1150   0684 1151   0687 1152   068A 1153   0690 1154
0693 1155   0699 1156   069F 1157   06A2 1158
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____